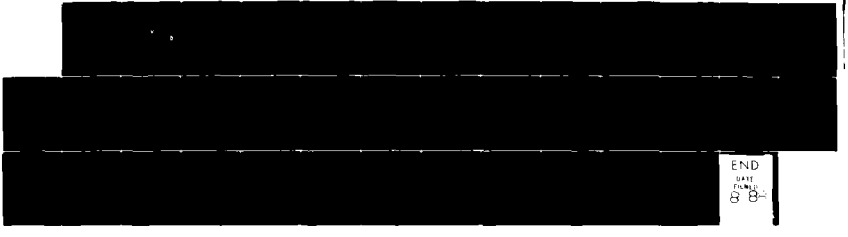
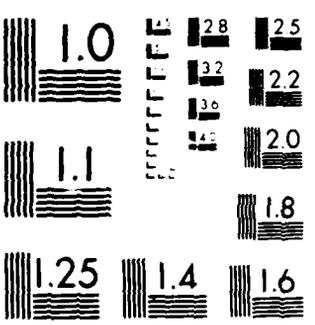
AD-A195 190 OPTIMAL TRANSFER LOTS FOR BATCH MANUFACTURING: A BASIC CASE AND EXTENSIONS(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA D TRIETSCH NOV 87 NPS-54-87-010 1/1

UNCLASSIFIED F/G 13/8 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD-A195 170

DTIC FILE COPY

NPS-54-87-010

NAVAL POSTGRADUATE SCHOOL

Monterey, California

DTIC ELECTE JUN 14 1988 D E

OPTIMAL TRANSFER LOTS FOR BATCH MANUFACTURING: A BASIC CASE AND EXTENSIONS

DAN TRIETSCH

November 1987

Approved for public release; distribution unlimited.

Prepared for: Naval Postgraduate School
Monterey, CA 93943

88

NAVAL POSTGRADUATE SCHOOL
Monterey, California

RADM. R. C. Austin
Superintendent

Kneale T. Marshall
Acting Provost

The research summarized herein was accomplished with resources provided by the Naval Postgraduate School.

Reproduction of all or part of this report is authorized.

This report was prepared by:

Dan Trietsch

Dan Trietsch
Adjunct Professor
Department of Administrative Sciences

Reviewed by:

David R. Whipple, Chairman
Department of Administrative Sciences

Released by:

James M. Fremgen
Acting Dean of Information and Policy Science

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

ADA195170

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION Unclassified	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-54-87-010	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable)	7a NAME OF MONITORING ORGANIZATION
6c ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code)
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
Optimal Transfer Lots for Batch Manufacturing: A Basic Case and Extensions (UNCLASSIFIED)

12 PERSONAL AUTHOR(S)
Dan Trietsch

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Final Report	FROM _____ TO _____	November 1987	35

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Batch Production; Transfer Lots; Minimal Makespan; MRP

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

One of the weaknesses of most MRP systems, is that they do not allow overlapping of activities on a batch by sequential machines. However, by letting the second machine start processing a batch before the first one has finished it yet--a well known idea utilized by many practitioners--we can often achieve considerable improvement in terms of shorter lead time. This requires the use of partial transfer lots. In this paper we address the problem from a theoretical point of view. We optimize the sizes of the transfer lots for overlapping processing on two machines, so as to minimize the makespan under a constraint on the number of transfers. We also discuss the issue of several consecutive jobs (batches), and how to extend the solution to several machines in such a manner that all machines process the batch continuously.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	Unclassified	
22a NAME OF RESPONSIBLE INDIVIDUAL Dan Trietsch	22b TELEPHONE (Include Area Code) 408, 646-2456	22c OFFICE SYMBOL 54Tr

DD FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

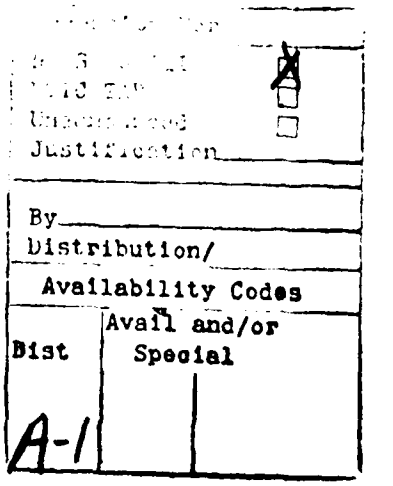
OPTIMAL TRANSFER LOTS FOR BATCH MANUFACTURING:

A BASIC CASE AND EXTENSIONS

By

Dan TRIETSCH*

November 1987**



---

* Department of Administrative Sciences, Naval Postgraduate School, Monterey, CA 93943.

** The first draft of this paper was written in February 1987.

Optimal Transfer Lots for Batch Manufacturing: A Basic Case and Extensions

Abstract

One of the weaknesses of most MRP systems, is that they do not allow overlapping of activities on a batch by sequential machines. However, by letting the second machine start processing a batch before the first one has finished it yet--a well known idea utilized by many practitioners--we can often achieve considerable improvement in terms of shorter lead time. This requires the use of partial transfer lots. In this paper we address the problem from a theoretical point of view. We optimize the sizes of the transfer lots for overlapping processing on two machines so as to minimize the makespan under a constraint on the number of transfers. We also discuss the issue of several consecutive jobs (batches), and how to extend the solution to several machines in such a manner that all machines process the batch continuously.

1. Introduction

In the field of computerized decision support systems for production, two major products seem to be at the center of attention: **MRP** (I and II) [7; 6, pp. 655-658]--offered by many vendors--and **OPT** (Optimized Production Technology), a proprietary "black box" system [3; 2, pp. 607-619; 6, pp. 692-715]. In fact, OPT is not a totally different product than MRP, but rather a system designed to avoid some pitfalls which MRP users tend to fall into [10]. In a nutshell, MRP is basically a good book-keeping system with very crude planning capabilities, while OPT is touted by its vendor to be a full fledged model-base decision support system, capable of making so-called optimal decisions.

MRP is designed to calculate order release dates for components, so that the production of required end products will be accomplished on given due dates. In order to do so, the system assumes that there is a known fixed lead time between an order release and its completion. Using this logic, it schedules all activities backwards from the due dates. The result is often an infeasible schedule in the sense that resources are overloaded. Therefore, the planner has to adjust the lead times and/or the due dates, and try again. An implicit assumption behind that approach is that queuing requires a pseudo-deterministic time, and if this time is approximated correctly in the lead time assessment, things will fall into place adequately. This implicit assumption is the cause of a lot of grief to MRP users in practice. OPT proponents argue [3], and this author concurs, that the bulk of the variability in the queuing time in MRP driven systems can be eliminated by proper sequencing, scheduling and monitoring procedures, applied to critical activities (while still taking care of the inherent variability in the processing time and unforseen delays by using appropriately placed time buffers).

When a batch has to be processed on several machines sequentially, the

usual "MRP logic" is that the first machine has to finish the whole batch before the second machine can start. This in turn means losing opportunities available by overlapping the activities. This is accomplished by processing the first part of the batch on the second machine while the first machine processes the second part, etc. (see Figure 1). OPT allows for overlapping activities, by employing proprietary procedures to plan transfer lots.

OPT is designed to make the scheduling decisions automatically. Part of the output consists of a list of activities which have to be performed as soon as possible. These activities are bottleneck activities. In order to meet the due dates, the bottleneck activities have to meet or beat the timetable which OPT generates. Another part of the output is a list of non-bottleneck activities. To avoid excessive work in process inventory, they should not be performed before schedule. The planner's role is limited to entering data. There is no interactive capacity, and no questioning of the "optimal schedule" is allowed.

In addition to these two computerized decision support systems, we hear a lot about **JIT** (just-in-time) [2, pp. 714-744] and about **GT** (group technology) [2, p. 718; 6, pp. 660-661]. These are not decision support systems, but rather production flow schemes designed to improve the operations of manufacturing plants. (Usually JIT users implement GT as part of their overall design, but GT can also be used without JIT.)

JIT, also known as **The Toyota Method**, is a pull system designed to reduce work in process inventory. JIT was designed primarily for assembly line (i.e., mass production) environments. Ideally, under JIT the items we produce are transferred one by one from station to station. The line makes these transfers very inexpensive to achieve. Parts required for the assembly are also fed to the stations which use them by very small lots, triggered by the usage. Setups,

are rigorously reduced to make the shift from one item or model to another as painless as possible--this is obviously a must if we want small batches. JIT also incorporates quality assurance and workers participation methods (**Quality Circles**) which are outside the subject matter of this paper.

GT is a plant layout and organization scheme. The machines (or other resources) are arranged in such a manner that typical products flow along one of several prearranged routes in a "line-like" manner. The chief advantage of GT, relative to a regular job shop arrangement, is reducing the cost of transferring items or lots from one machine to the next. This makes it possible to use small transfer lots, and thus operate in a "JIT like" manner.

Organizing the plant according to the GT concept need not have much influence on the decision whether or not to use MRP or OPT. With JIT we may use a basic MRP computerized system to handle floor shop order releases and purchasing orders. However, by choosing JIT, the scheduling decisions OPT is designed to make are made manually by trial and error while operating the plant. Therefore, in practice, those systems are in competition with each other. In theory, much of the "OPT philosophy" was originally developed by the JIT people.

The basic premise of this paper is that MRP may be an adequate production book-keeping system, but it leaves a lot to be desired as a decision support system for scheduling and planning. On the other hand, the "OPT philosophy" is good for many situations, especially for batch production (i.e., medium volume production). A major benefit is a reduction the expected value of and the variability of the lead time, which in turn implies less work in process inventory and better performance in terms of on-time deliveries. However, the black box approach is rigid and the lack of interactive capacity does not sit well with human needs and motivating creativity in the work place. Also, many sophisticated users tend to distrust the system, since it is based on secret

procedures, which most probably are simple heuristics. In conclusion, neither MRP nor OPT are satisfactory decision support systems for scheduling today.

The solution? One option is to develop MRP to handle the scheduling problem better than it does today, and in such a manner that it will still be possible to use it interactively. In other words, develop MRP to be better than today's OPT. The algorithms used by such an enhanced MRP system should be in the public domain. This would allow all vendors who wish to use them to do so. Furthermore, making the system "open for inspection" would make it easier for consultants to recommend it to their customers. It would also make it easier to tailor the system to the customer's needs. This paper is designed to provide some of the theoretical background required to that end.

Basically there are three things OPT does which MRP does not do [10]: (i) identify bottleneck resources; (ii) schedule activities on bottlenecks (and downstream from bottlenecks) forward instead of backwards, thus utilizing them fully; and, (iii) allow transfer lots to be smaller than the batches they belong to, thus making overlapping processing on sequential machines possible.

Much has already been published elsewhere to make the implementation of (i) and (ii) possible (though one would often have to resort to heuristics). For instance, linear programming can be used not only to identify bottlenecks--better known as binding constraints in LP parlance--but also to optimize the product mix at the same time. Some work has also been published in the realm of (iii), but under restrictive assumptions. A recent example is [4], and the interested reader may find references to earlier efforts there. The assumptions in [4] are constant demand and identical production rates for all machines, and the model is developed for two machines. The model optimizes the number of transfer lots under the assumption that they should be equal and integral. In case more machines are involved the authors suggest to apply their model on a

pair by pair basis. As we shall see below, if we assume equal production rate then the optimal transfer lots should indeed be equal to each other, so this assumption is not strong. However, as the authors indicate, the assumption of equal rates itself is strong. In conclusion, it seems that (iii) still justifies further efforts at this stage, in particular by (1) lifting the equal production rate assumption, (2) considering the interaction among several jobs on the same set of machines in some detail, and (3) dealing with several consecutive machines. In this paper we deal with (1) and (2). As for (3) we use a pair by pair procedure as in [4], but suggest a fast heuristic which helps to allocate the transferring efforts to consecutive pairs. Results which are superior to the pair by pair method are presented in [9]. Also, it has recently been brought to this author's attention that some of these issues are being tackled independently by Baker [1], who obtains some (but not all) of the results of this work and [9] by a significantly different approach.

In this paper, we analyze the two machines case under the assumption that the batch size is given (e.g., by one of the existing models), and the number of transfers is limited by a budget constraint. We concentrate on planning the exact optimal (and not necessarily equal) transfer lots. The algorithm we obtain can be used either forward or backwards very easily. Incorporating it in any existing MRP package should be quite simple.

The rest of the paper is organized in six sections. In Section 2, we present the problem formally and discuss its parameters. In Section 3 we solve a simplified version of the two machines case where fractional items can be transferred. This is obviously a relaxation, and it will serve us later as a basis for a more realistic version, discussed in Section 4, where the transfer lots are restricted to be integral. In Section 4 we also discuss the issue of accommodating scheduling backwards instead of forward, which makes it more

convenient to apply in existing MRP systems. Section 5 is devoted to a sensitivity analysis. Section 6 discusses the issue of several jobs and presents the heuristic for allocating the budget to consecutive pairs of machines. When used in conjunction with optimizing the budget itself, our heuristic is optimal. Finally, Section 7 is the conclusion.

2. The Basic Problem

We denote our basic problem, with n machines, as (P):

(P) *A batch of m items needs to be processed sequentially on n machines,* M_1, M_2, ... , M_n. Each item requires T_i time units of processing on M_i; i = 1, 2, ... , n. In addition to the marginal processing time T_i, M_i requires a setup time of SU_i; $\forall$ i. Transferring a batch of any size (up to and including *m* items) from M_i to M_{i+1} costs C_i money units, and takes TT_i time units; i = 1, 2, ... , n-1. *It is required to finish the production run on all machines in* minimal time (i.e., minimize the make-span), subject to a budget constraint on the total transferring cost, B. ●

Obviously, specifying the sizes of each lot or sub-lot for each machine can be done in exponential time. Given these sizes, the optimal scheduling of the machines can be done by stipulating that they will process each item as soon as it reaches them or when they are free, whichever is later. Therefore, (P) can be solved completely by algorithms in NP. However, the solution above is not tractable in practice, so more efficient methods are required.

We denote the special case where n = 2 as (P1). In the rest of this paper we restrict ourselves to (P1) and its derivatives. As a convention, we let (Si)

denote the solution to (Pi). E.g., (S1) is the solution of (P1), and (S) is the solution to (P) itself. For convenience we suppress the indices of C_1 and TT_1, and any other index which can only assume a single value.

Ignoring the budget for a while, it may be argued that all we have to do is transfer the items in the batch one by one. This is indeed the idealized JIT approach, where in a sense we have one big super-batch, and if possible we transfer the items one by one. However, in addition to the cost issue itself, there are two reasons why this may not be the best thing to do. First, as we'll show below we can often achieve the same results in terms of minimizing the makespan with less then m transfers. Second, our transferring frequency may be constrained (in this case there exists a budget which will prevent us from violating this constraint).

Since we do use a budget in our formulation, it would be nice if our model would also help us to determine the optimal budget which minimizes the total costs of transfers vs. lead times savings. Indeed our model will enable us to optimize for the best equilibrium between the transferring expenditure and the time saved by the procedure.

Are there any circumstances under which the model will lead to transferring the items one by one? If the transfers are very inexpensive our procedure will lead us to specify small transfers. If, in addition, the machines are balanced (i.e., have the same production rate), then the model will indeed indicate transfer lots of one, as in idealized JIT. Otherwise, the transfer lots are not equal in general, and even if we do move the first or last item(s) one by one, other transfer lots will tend to be larger.

For a batch job shop environment this may be the closest we can get to JIT. This is due to the fact that transfers do not tend to be as easy in a job shop as they are in the JIT environment, since the routing of the batch may be

haphazard (here is where GT comes in!). By taking the transfer costs into account, as we do, we can strike an optimal balance between "JIT" (or transferring one by one) and "MRP" (transferring the whole batch together).

To continue, note that in the two machines case--in contrast to the several machines case--we can express the budget in terms of the maximal number of possible transfer lots, which we proceed to do. W.l.o.g., this number may be restricted to be m at most. Below we show that it will often be o(logm).

It is intuitively clear that in order to minimize the completion time we should start the batch on M_2 as soon as possible, provided that we can then process the whole batch on M_2 continuously. This pushes us in the direction of using as many transfers as the budget allows, thus making them smaller on average, and allowing M_2 to start earlier.

Note that if $T_1 > T_2$, then M_1 cannot feed M_2 continuously unless the first transfer lot is large enough; in that case we gain nothing by transferring the first lot too soon (and thus making it smaller than optimal).

On the other hand, if $T_1 < T_2$, then it may happen that while M_2 is processing the first small transfer lots, M_1 can produce larger and larger ones without starving M_2. Again, this may call for less transfers than the maximum the budget allows.

Finally, in both cases, it may happen that SU_2 places a binding constraint on the time when we can start processing on M_2. In this case, the first transfer batch should never be smaller than the maximal number of items which can be processed on M_1 and transferred to M_2 before M_2 is ready.

3. Some Interim Results

In this section we solve a relaxed version of (P1) where it is allowed to transfer fractions of items, as long as each transfer lot is not smaller than one item. We begin with the simplest case, namely, where we ignore the setups and the transfer time and allow exactly one extra transfer (i.e., B = 2C). We designate this special case of the problem as (P2), and its solution as (S2). Next we generalize (P2) by allowing up to m-1 extra transfers, i.e., for any B. Then we show that the marginal gain by each successive transfer is monotone decreasing. Finally, we show how to deal with lower bounds on the first transfer, e.g., due to SU_2. In Section 4 we show how to adapt the solution of the relaxed version to the unrelaxed problem.

(P2) Solve (P1) under the following assumptions: (i) $SU_1 = SU_2 = 0$; (ii) exactly one extra transfer is allowed (i.e., B = 2C); and, (iii) fractional items may be transferred, but never less than one item in each lot. ●

(S2) Let $0 < L_1 \leq 1$ be the size of the first transfer lot, and $L_2 = m - L_1$ be the relative size of the second, final, transfer lot. Our problem is to solve for L_1 and L_2. If we choose L_1 and L_2 in such a manner that M_1 requires the same time to process the second lot as it takes for M_2 to process the first one, then

(1) $$L_2T_1 = L_1T_2.$$

Define $Q_i = T_{i+1}/T_i$, and suppress the index of Q_1 for (P1), then

(2) $$L_2 = L_1T_2/T_1 = L_1Q.$$

But $L_1 + L_2 = m$, and hence

(3) $$L_1 = mT_1/(T_1 + T_2) = m/(1 + Q).$$

We now prove by negation that L_1, L_2 as above are optimal. We do this

under the assumption that they do not violate the bounds $L_1, L_2 \geq 1$. First assume that we transfer less than L_1 as the first batch. In this case, M_2 would finish processing the first transfer before M_1 finishes the rest of the batch, and will then have to wait till time mT_1 for the second batch, but this second batch will be larger than L_2 as above, and hence will require a longer time to finish than under our scheme. Next assume that we transfer more than L_1 as the first lot, then M_2 will start operating later then before; but under our scheme it is possible for M_2 to operate continuously, so starting later means finishing later!

If L_1, L_2 do violate one of the constraints $L_1, L_2 \geq 1$, then the violated bound dictates the solution. For instance, if $L_2 < 1$, then the first lot will have m-1 items, and the second lot will have one item (we omit the proof). Note that unless $T_1 = T_2$ (i.e., $Q = 1$), the transfer lots are not equal to each other, as mentioned above. ●●

(P3) Solve the single batch two machine case where $1 \leq k \leq m-1$ extra transfers are allowed; and fractional items may be transferred, but never less than one item in each lot. ●

(S3-Part 1) In this part of the solution we assume that no transfer lot of less than one will occur. We also assume $SU_1 = SU_2 = TT = 0$. Later, in (S3-Part 2) we'll show the appropriate correction if this is not really the case. Under these assumptions, the time required by M_1 to process the j^{th} lot should equal the time required by M_2 to process the $(j-1)^{th}$ lot, for any $2 \leq j \leq k$. The proof follows the one given in (S2) inductively, and we omit the details. If we designate the size of the j^{th} lot by L_j, then

(4) $$L_jT_1 = L_{j-1}T_2, \; ; \; 2 \leq j \leq k,$$

(5) $\quad L_j = L_{j-1}Q = L_1Q^{j-1}; \quad 2 \leq j \leq k.$

But $L_1 + L_2 + \ldots + L_k = m$, and using the familiar formula for the geometric progression we get

(6) $$L_1 = \begin{cases} m(1-Q)/(1-Q^k) & ; \quad Q \neq 1 \\ m/k & ; \quad Q = 1. \end{cases}$$

Obviously (S2) is a special case here. ■■

How much time can we actually save by using this optimal solution? Since our procedure allows for continuous processing on M_2, the time we save relative to the regular MRP (single transfer) case is $T_1(m - L_1)$. Obviously, by driving L_1 as low as possible, we maximize our savings. As long as the budget allows and all transfer lots are strictly larger than one, we can always make L_1 smaller by adding a transfer; this is trivially easy to show when fractional items can be transferred, as is the case here. Still, the maximal possible gain is limited. In fact, the best we can hope for is for the faster activity to process m-1 items in parallel to the slower one, thus saving most of the time required for the faster one. For instance, by transferring each item separately, regardless of whether or not M_2 is ready, we achieve the maximal gain by exactly m transfers. It is easy to verify that the makespan in this case will be $m\text{Max}\{T_1, T_2\} + \min\{T_1, T_2\}$, and by subtracting this from $m(T_1 + T_2)$ we obtain the maximal possible gain (MPG), and without violating the integrality constraint, as follows

(7) $\quad \text{MPG} = (m-1)\min\{T_1, T_2\}.$

We also need to know what is the marginal gain by the k-th transfer, denoted by MG(k). Obviously MG(k) is simply the difference between L_1 for k-1 and for k transfers. Assuming $k \geq 2$ (and substituting $1 = Q^0$ if necessary) we obtain:

(8) $MG(k)/T_1 = L_1|k-1 \text{ transfers} - L_1|k \text{ transfers} =$

$m/(1+Q+Q^2+\ldots+Q^{k-2}) - m/(1+Q+Q^2+\ldots+Q^{k-1}) =$

$mQ^{k-1}/[(1+Q+Q^2+\ldots+Q^{k-2})(1+Q+Q^2+\ldots+Q^{k-1})]$; or,

(9) $MG(k) = mT_1Q^{k-1}/[(1+Q+Q^2+\ldots+Q^{k-2})(1+Q+Q^2+\ldots+Q^{k-1})]$.

It is interesting to note that the first extra transfer (i.e., going from one to two transfers) gives us more than half of the total possible savings under (P3). To see this note that by (S2) we save $mT_1T_2/(T_1+T_2)$. W.l.o.g. assume $T_1 \geq T_2$, then by (6) our maximal savings is $(m-1)T_2$. But clearly $mT_1T_2/(T_1+T_2) \geq mT_2/2 > (m-1)T_2$ in this case.

Since more than half of the potential is already used up, the second extra transfer cannot be as beneficial as the first. In fact, the next theorem tells us that no extra transfer can be as beneficial as the preceding one, even though they continue to bring about positive gains as long as no lot is less than 1.

Theorem 1: The marginal gain MG(k) is monotone decreasing with k (i.e., we have convexity!), and non-negative.

Proof: The non-negativity is trivial. To show the convexity, first assume $Q \leq 1$ (recall $Q = T_2/T_1$, hence this means M_1 is not faster than M_2). Under this assumption the numerator in (9) is monotone non-increasing and the denominator is monotone increasing with k. Hence, $MG(k) < MG(k-1)$ as required.

To prove for $Q > 1$, we observe that the two cases, $Q < 1$ and $Q > 1$ are symmetric in the following sense: the first calls for decreasing transfer lots, and the second for increasing ones; but if we reverse the sequence, thus having the second faster machine feeding the first slower one, i.e., $Q < 1$ again, we obtain identical lot sizes to the ones we had before, though in reversed sequence. It follows that we also have the same total production time and thus the same savings by adding a transfer. This completes our proof. ■■■

It follows that if for some k the marginal gain by the k-th transfer does not justify the marginal expense, C, then it will certainly not be worthwhile to transfer more than k extra lots. In other words, this allows us to optimize the budget very easily.

If we need to actually calculate MG(k)--to determine if another transfer is worth the expense, for instance--then by using the geometric progression formula and (9), we get

$$(10) \qquad MG(k) = \begin{cases} mT_1Q^{k-1}(1-Q)^2/[(1-Q^{k-1})(1-Q^k)] & ; \ Q \neq 1 \\ mT_1/[k(k-1)] & ; \ Q = 1. \end{cases}$$

As was the case for (8) and (9), we assume $k \geq 2$ here.

(S3-Part 2) We are now ready to deal with lower bounds on L_1. These may occur due to SU_2, or if L_1 turns out to be less than one under (S3-Part 1). A similar, symmetric, approach can be used to ensure that the last lot will be at least one as well. In both cases we may find that this may imply less transfers than budgeted for, as discussed above. Note that if the bound is due to SU_2, we do not require yet that it be integer.

To continue, The bound on L_1 due to SU_2 is

$$(11) \qquad L_1 \geq (SU_2 - SU_1 - TT)/T_1.$$

And, in order to guarantee that the first lot will be of size one at least, we require $L_1 \geq 1$. Using the notation [(i)] for the numerical value obtained by applying (i), we now have the following expression for L_1:

$$(12) \qquad L_1 = \text{Max } \{[(6)], (SU_2 - SU_1 - TT)/T_1, 1\}.$$

If (12) is not satisfied by [(6)], we can use a very simple approach. We take L_1 to be as per [(12)], and calculate L_2, L_3 etc. iteratively by (5). As we do this we check if the sum does not yet exceed m; when it does, we adjust the size of the last lot to m less the sum of the others, and stop.

Looking at this procedure, we find we can improve it by calculating in advance how many lots will be required. The result is also useful for the purpose of specifying the maximal useful budget. Assume first that $Q > 1$, then (5) leads to increasing lots, therefore by specifying L_1 and using (5) for the other L_j values we obtain an increasing geometric series. The question is: how many members should the series have in order to exceed m for the first time? We denote the answer by K:

(13) $\quad K = \text{SUPINT}(\log[(Q - 1)m/L_1 + 1]/\log Q) \quad ; \quad Q > 1,$

where SUPINT(x) = smallest integer $\geq x$. Next, if $Q < 1$, then

(14) $\quad K = \text{SUPINT}(\log[(1/Q - 1)m/L_1 + 1]/\log[1/Q]) \quad ; \quad Q < 1.$

Finally if $Q = 1$, then $K = m$.

By examining the result, we can also state that unless $Q = 1$ we only require $o(\log[m/L_1]) \leq o(\log m)$ transfers to achieve any feasible gain!

Furthermore, L_K should be ≥ 1 as well (when both L_1 and L_K are 1 at least, then the same holds for all the other L_j values). If it is not, then we can reverse the order of the calculation in (5), and using $L_K = 1$ calculate the other L_j values by the following recursion

(15) $\quad L_j = L_{j+1}(1/Q) \quad ; \quad j = K-1, K-2, \ldots, 1.$

At this stage, we can also see that if we were to use K-1 or less lots, then the bound would not be binding, and L_1 would again be determined by (6).

Note also that if $L_1 \neq [(6)]$, than the solution implied by the procedure described above, though optimal, is not unique. This is due to the fact that in such cases we have some degree of freedom in adjusting the downstream lots. Also, it is possible in such cases that the relaxed makespan will equal the real minimal makespan obtainable with integral L_i's.

4. The Formal Solution

Obviously, in order to solve (P1), we have to solve (P3) with the additional constraint that all transfer lots will comprise integral numbers of items. As we shall see, bounds on L_1 will not require additional special treatment beyond that discussed above due to this additional requirement. Therefore, we ignore such bounds in the following analysis.

First let us introduce a straightforward procedure designed to feed M_2 continuously without breaking the integrality constraints. We will refer to this procedure as **the integrality procedure.** Not surprisingly, the integrality procedure assumes that we allow enough extra time relative to the (S3) schedule. The output of the integrality procedure is a feasible solution to (P1).

The basic premise of the procedure is that since we allow ourselves to lose some time, say h, relative to (S3), then there is no need to start processing on M_2 before time $SU_1 + TT + L_1T_1 + h$ (under (S3) we start at $SU_1 + TT + L_1T_1$). On the other hand, if we wait till that time to start, it should be possible to feed M_2 continuously from then on till completion. In other words, if h is a feasible delay it should be possible to make all lot deliveries required by (S1) at the time M_2 completes the processing of the previous ones. Furthermore, this should be possible without exceeding the budget. A feasible scheme for doing this would be to make the first transfer at $SU_1 + TT + L_1T_1 + h$, and then as soon as M_2's buffer drops below the amount necessary to feed it for TT time units, transfer immediately all the items which M_1 finished processing by that time.

Clearly the procedure cannot fail if h is feasible. If we try it with an infeasible value, however, we either fail to make a delivery in time (if the budget remaining is needed for the last lot but M_2 finished the next to last one), or we require more transfers than stipulated to finish the batch.

Next, we show a simple bound on the maximal extra time which may be required. This result is presented as a theorem. The proof, even though almost trivial, is a constructive one, and as a result we obtain two simple heuristics designed to adapt (S3) to the integrality constraint in such a manner that the bound is not violated.

By way of preparation, let us define S_i as the cumulated number of items in the first i lots, i.e.

$$(16)\quad S_i = \begin{cases} \sum_{j=1,i} L_j & ;\ i = 1, 2, \ldots, m \\ 0 & ;\ i = 0. \end{cases}$$

And let f_i be the fractional part of S_i. Further, let e_i be

$$(17)\quad e_i = \begin{cases} 1-fi & ;\ f_i > 0 \\ 0 & ;\ f_i = 0. \end{cases}$$

Note that $f_i = e_i = 0$ if and only if there is no fraction.

Theorem 2: Let h^* denote the difference between the completion times under (S3) and (S1), then

$$(18)\qquad h^* \leq \min\{\max_i\{e_i T_1\},\ \max_i\{f_i T_2\}\} \leq \min\{T_1,\ T_2\}.$$

Proof: We prove the theorem constructively by introducing a pair of heuristics designed to achieve feasible solutions to (P1) with respective losses of $\max\{f_i T_2\}$ and $\max\{e_i T_1\}$. The first heuristic is to round all the S_i values up to the nearest integer, i.e., S_i+e_i, and use these values as the real cumulated lot sizes. For instance, the new value for L_i will be $(S_i+e_i) - (S_{i-1}+e_{i-1})$. These lot sizes certainly are feasible under (P1), and they do not require more transfers than (S3). (Note that Sm = m, is integral and requires no truncation.)

Now, if we start processing under this new scheme, the most M_2 will have to wait for a lot due to its being too large, and thus not ready on time as per

(S3), is $\text{Max}\{e_i T_1\}$, therefore we have a feasible solution with that value.

It remains to show a similar method which achieves a loss of $\text{Max}\{f_i T_2\}$. This can be done by truncating all the S_i values down to the nearest integer, i.e., $S_i - f_i$. We prove this using symmetry: If we change the order of the machines (S3) will retain the same lot sizes, only in reversed order; i.e., the *original* S_i would become $m - S_{m-i}$ etc., T_1 and T_2 will trade places, and the e_i values would become the f_i values of the new reversed S_i values and vice versa. Applying the rounding up heuristic to the reversed problem is identical to using the truncating algorithm on the original, and the total processing time will remain unaltered. Therefore the difference between (S1) and (S3) will be unaltered as well. ■■■

At this stage, using the integrality procedure and the bound, we can apply simple search methods to approach h^*, the minimal loss, as much as we wish. For instance, by halving the search area iteratively.

However, there is a way to find the exact optimal delay, which also makes use of the integrality procedure. Assume we choose h too small, and hence it is infeasible, then at least at one stage, M_1 failed to deliver the last unit of a lot, required under (S1), in time. If we start raising h slowly, there will come a moment when somewhere one of the lots defined by the procedure will jump by one. This may or may not suffice to achieve feasibility. If it is, we're through because the minimal h value yielding feasibility is our optimum; otherwise, all we have to do is to continue raising h till another lot jumps up, and so on. Thus we only have to check a finite number of potential increases in h, and stop when we achieve feasibility. We'll refer to this as **the optimizing procedure.**

In order to implement the optimizing procedure technically, we start with $h_0 = 0$, and we are in the first iteration. In the k^{th} iteration, we have h_{k-1}. Using h_{k-1} with the integrality procedure, we check if a feasible solution has been obtained. If so, then $h^* = h_{k-1}$, and we're through. Otherwise, we list all the fractional items being processed on M_1 at the transfer times. Denote the fraction completed on the item being processed on M_1 when the i-th lot is transferred by F_i, and let

$$E_i = \begin{cases} 1-F_i & ; \ F_i > 0 \\ 0 & ; \ F_i = 0. \end{cases} \tag{19}$$

Then, we increase h by $\min\{E_i\}T_1$, i.e.,

$$h_k = h_{k-1} + \min_i\{E_i\}T_1, \tag{20}$$

and start the (k+1)-th iteration.

Note that we could start the optimizing procedure with any h_0, not necessarily $h_0 = 0$, if we knew somehow that $h_0 \leq h^*$. This raises the option of first using a search method as described above to reduce the search domain. Our objective is to identify candidate values for h_0 which are more than 0. We may also wish to reduce the search domain sufficiently to enhance the possibility that the optimal search will terminate fast. After the initial search, we start the optimizing procedure with the highest non-feasible h value as h_0. If we choose to do that, we may want to tune the program so that the total number of iterations in both modes will tend to be as low as possible.

In practice, a simple program was developed for the optimizing procedure to gain some computational experience, and it turns out tnat the procedure is very fast. The number of iterations required by it even with $h_0 = 0$ is usually well below a third of the number of transfers allowed. In most instances, h^* was less than half of the bound as per Theorem 2, $(\min\{\text{Max}\{e_iT_1\}, \text{Max}\{f_iT_2\}\})$. Therefore, it seems that the optimality procedure does not require further

improvements such as combining a preliminary search as described above.

If the non-integral solution, (S3), violates a bound on L_1, our procedure was to set L_1 exactly as per the bound. As mentioned above, under these circumstances it is quite likely that the integrality constraints can be maintained without further delays. In this case the optimality procedure will terminate with $h^* = h_0 = 0$.

Another issue we need to discuss is using the algorithm for scheduling backwards. This is important for MRP systems, since they usually do schedule backwards. In this case, if we interchange the order of the machines and assume $SU_1 = SU_2 = 0$, the algorithm will be applicable. Of course, the order of the transfer lots will be reversed in this case. It remains to discuss the case where the setup times are not zeros.

In order to take the setup times into account, we have to find a bound on our new "first" transfer lot. Rather than do this from scratch, we can use the bound we already have for scheduling forward, and translate it to the one we need now. First note that the total processing time is the time required to process L_1 items on M_1, L_1T_1, plus the time required to process the whole batch on M_2, mT_2. But the total processing time is unaltered under the reversed order. Therefore, using symmetry we obtain

(21) "old-bound"$T_1 + mT_2$ = "new-bound"$T_2 + mT_1$, or

(22) "new-bound" = $m - (m -$ "old-bound"$)T_1/T_2$;

where "old-bound" and "new-bound" are the original bound on L_1--as per (11)--and the new bound on the last transfer lot (which takes the role of the first lot now), respectively.

An Example

We conclude this section with an example, depicted in Figure 1. Let $m = 120$, $T_1 = 2$, $T_2 = 3$, $C_1 = 5$, $B = 26$. Therefore, $q = 1.5$ and $k = 5$. Applying the relaxed solution we obtain $L_1 = 9.1$, $L_2 = 13.65$, $L_3 = 20.47$, $L_4 = 30.71$, and $L_5 = 46.07$. The makespan is 378.2. The integrality procedure is employed at this stage, and the minimal E_i value of 0.4 is located at the second lot, leading to $h_1 = 0.4 \cdot 2 = 0.8$, and a tentative makespan of 379. It turns out this is feasible, with $L_1 = 9$, $L_2 = 14$, $L_3 = 21$, $L_4 = 31$, and $L_5 = 45$. Therefore, this is the optimal solution here. Note that M_2 starts processing at time 19, though the first lot arrives at time 18, which gives us a slack of 1 for the purpose of transferring L_1. Similarly there is a slack of 1 for L_4 and a slack of 4 for L_5. The latter slacks can also apply to processing the lots, but if we delay processing L_1, L_2, or L_3, the makespan will increase. Since we used integral T_i values, it is clear that the optimal makespan has to be integral too, however, the integrality procedure does not require integral T_i values.

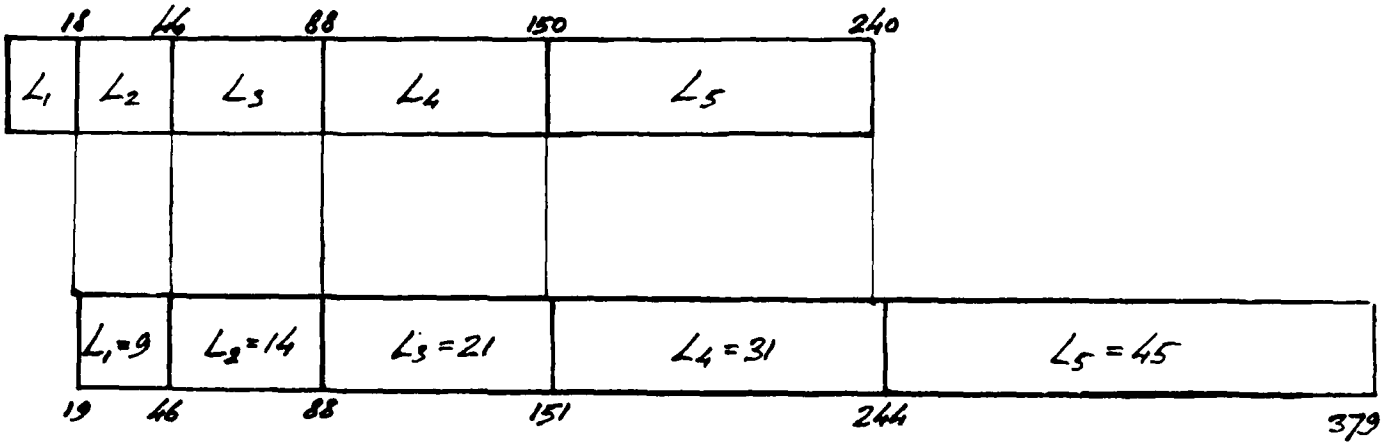


Figure 1

5. Sensitivity Analysis

Having computed the optimal solution for the two machines case, we are now ready to perform some sensitivity analysis calculations. We deal explicitely with (S2), and under the assumption that the SU_2 does not constitute a binding constraint on L_1 for the number of transfers, k, which are allowed. The objective of this section is to assess the impact of slight errors in the estimates of T_1 and T_2, as well as the impact of slight deviations from the plan in the execution stage. For the latter, we only consider the impact of increasing or decreasing L_1 relative to the optimal value computed by (6). Variations in latter transfer lots have a similar effect since one can always look at the last k-j transfer lots as an instance of the original problem with less items and a smaller budget. In addition, there is another reason why we choose to analyse the sensitivity of the performance to variations in the size of the first transfer lot, as elaborated below.

When expediting a batch, many floor managers practice a slight variation of our integrality procedure. That is, they transfer a partial transfer lot to M_2, and then continue to feed it by additional transfer lots as necessary to ensure its continuous operation (if possible). We refer to this practice as **immediate feeding**. The only difference between the immediate feeding practice and using our model is that the size of the first transfer lot is not calculated exactly, and is not optimized in conjunction with the transfer expenses. By performing a sensitivity analysis on the first transfer lot size then, we obtain a measure of the contribution of our model in these cases. (This is in addition to the fact that our model impacts the planning process, while the immediate feeding practice does not.)

If the first transfer lot is slightly too large, i.e., the transfer is delayed by some time, then M_2 will be equally delayed. There is no way later

actions can change this (though additional delays can be added of course). The case is more complicated if the first transfer lot is too small. In this case, if the floor manager follows the immediate feeding practice for the downstream transfers, the subsequent transfer lots will be proportionally smaller too, and if we wish to keep M_2 busy continuously, then at least one additional transfer will be required above the budget allowance. The number of additional transfers required is a monotone step function of the amount by which the first transfer lot is too small. Now, if we take the stance that additional transfers above the budget are strictly prohibited, then the result of the procedure will be that M_2 will have to wait between the $(k-1)^{th}$ and k^{th} transfer, since this is when the budget for extra transfers (not counting the last and absolutely necessary transfer) will be depleted. In this case the relative error in the size of the first transfer lot is compounded, and in general the makespan can be very sensitive to this type of error when combined with the immediate feeding operating procedure. On the other hand, if we take corrective action starting with the second transfer (in other words, we follow the optimality criterion of Bellman), then the makespan will be affected much less. Finally, the floor manager may choose to return to the original plan starting with the second transfer lot, without regard as to whether or not M_2 is starved (in other words, the deviation is simply ignored). In this case the last transfer lot will have to be larger than planned, to compensate for the first lot being too small; as we shall see, this policy yields better results than the immediate feeding policy, and it is clearly much easier to apply practically than taking optimal corrective action (which requires updating the data and recalculating the model during the processing of the first lot).

We introduce the superscript o to denote the optimal values, and the superscript a to denote the actual values. For example, L_j^o and L_j^a are the

computed optimal and actual sizes of the j^{th} transfer lot respectively. We denote the makespan by MS, i.e.,

(23) $MS^o = SU_1 + TT + L_1^oT_1^o + mT_2^o$.

For the purposes of the present sensitivity analysis we assume $T_i^o = T_i^a$, and drop this superscript. (Note that L_1^o is a function of $Q^o = T_1^o/T_2^o$, and hence the impact of errors in estimating T_i^o is more profound than a casual inspection of (23) may reveal.) Our concern now is with the derivative of MS as a function of L_1 only; in other words, we assume the budget is enforced strictly. We leave it to the reader to investigate how many additional transfers are called for as a step function of the deviation in L_1 when it is too small and extra transfers are tolerated. In that case the makespan is improved of course, but if that advantage is valuable enough to offset the extra expense, then the budget was too tight to begin with.

Since the mechanism by which the makespan grows as a result of deviations in L_1 is different for the case where L_1 is too large and the case where it is too small, the derivative we compute is directed, and does not exist for a deviation of zero. We express the derivative as the rate in which the makespan grows as a function of the absolute deviation $\Delta L_1 = |L_1^a - L_1^o|$, and since the minimum is obtained for L_1^o, it will be positive in both directions. If the first transfer is too large, i.e., $L_1^a - L_1^o > 0$, the directed derivative is simply T_1. In the other direction, we calculate separately for (i) the immediate feeding scenario where the error is allowed to compound, (ii) the scenario where we stick to the original plan (and thus have the last transfer compensate for preceding deviations), and (iii) the scenario where optimal corrective action is pursued starting with L_2. Starting with (i), the sum of the first k-1 elements in this case is

(24) $(m - L_k^o)L_1^a/L_1^o$.

Therefore, the size of the last transfer lot will be:

(25) $L_k^o + \triangle L_1(m - L_k^o)/L_1^o$.

And the directed derivative is

$$(26)\quad dMS/dL_1|_{\text{compounded error}} = \begin{cases} T_2(m - L_k^o)/L_1^o & ; \quad L_1^a - L_1^o < 0 \\ T_1 & ; \quad L_1^a - L_1^o > 0. \end{cases}$$

Next consider (ii), where we stick to the original plan as far as possible, thus pushing the task of correcting former deviations to the last transfer lot. If $L_1^a - L_1^o > 0$, there is no difference in the end result of the former analysis. Otherwise, if $L_1^a - L_1^o < 0$, then M_2 will have to wait a bit for the second transfer lot after finishing the first lot. The time thus wasted will have to be added to the last transfer lot, and the result will be:

$$(27)\quad dMS/dL_1|_{\text{"original plan"}} = \begin{cases} T_2 & ; \quad L_1^a - L_1^o < 0 \\ T_1 & ; \quad L_1^a - L_1^o > 0. \end{cases}$$

Finally, let us consider the case of immediate optimal corrective action. In this case, starting with the second transfer lot, we are faced with an instance of (P2) with a budget for k-1 transfers. It is clear that the relationship between any two consecutive transfer lots should still be Q, and hence each of them must be multiplied by a constant such that their sum will be $(m - L_1^a)$. The appropriate constant is $(m - L_1^a)/(m - L_1^o) = 1 + \triangle L_1/(m - L_1^o)$, and since it applies to the k^{th} transfer lot as well, it implies a delay in the finishing time. This leads us to the following expression for the derivative:

$$(28)\quad dMS/dL_1|_{\text{optimal correction}} = \begin{cases} T_2 L_k^o/(m - L_1^o) & ; \quad L_1^a - L_1^o < 0 \\ T_1 & ; \quad L_1^a - L_1^o > 0. \end{cases}$$

Note that a similar strategy when $L_1^a - L_1^o > 0$, would not make a difference, since the second transfer lot would be smaller rather than larger, and would have to wait for M_2 to finish the former lot anyway.

It is easy to verify that for $k = 2$ there is no difference between (26), (27) and (28), while for $k \geq 3$ and $L_1^a - L_1^o < 0$ (28) yields a value which is strictly smaller than that associated with (26), while (27) leads to a value strictly between the former two.

Since the optimal correction strategy is probably not practical, we can conclude that the recommended course of action is to ignore former deviations and stick to the plan in the remaining operations. The theoretical value of (26) is by showing the potential disutility associated with the constant feeding practice when the first transfer lot is not optimized. The readers may note that in this case, if $T_1 > T_2$, then if the first transfer lot is small enough, it becomes impossible to feed M_2 continuously at all.

We are now ready to analyze the sensitivity to variations in Q. Here the assumption is that Q^o is not known exactly, and instead we use some estimate, Q^a, in our calculations. L_1^a is the result of the exact calculation but using Q^a instead of Q^o, and if we follow the original plan, so are the rest of the transfer lots. If we adopt some other rule for the subsequent transfer lots, then the situation will be similar to that described above, where $L_1^a \neq L_1^o$ for any other reason.

First let us analyze the case where the plan is stuck to throughout the whole batch. There are two mutually exclusive and exhaustive possibilities to consider. Either $Q^a > Q^o$, leading to $L_1^a > L_1^o$ and $L_k^a < L_k^o$, or $Q^a < Q^o$, leading to $L_1^a < L_1^o$ and $L_k^a > L_k^o$. In the former case, M_2 will not be through with the first transfer lot when the second one will reach it, and that pattern will continue till the end. In other words, the makespan will increase by $/\!\!\Delta L_1 T_1$. In the latter case, M_2 will have to idle between consecutive transfers, but the total makespan will still increase relative to optimality, since $L_k^a > L_k^o$. Let $/\!\!\Delta L_k = |L_k^a - L_k^o|$, then in the latter case the makespan will increase by $/\!\!\Delta L_k T_2$.

Using the chain rule we obtain

$$(29)\quad dMS/dQ|_{\text{"original plan"}} = \begin{cases} T_2|dL_k/dQ| & ; \quad Q^a < Q^o \\ T_1|dL_1/dQ| & ; \quad Q^a > Q^o, \end{cases}$$

where the following values should be plugged for the derivatives by Q

$$(30)\quad dL_k/dQ = \begin{cases} 0 & ; \quad Q = 1 \\ m(-k/Q^{k-1}+(k-1)/Q^k+1)/(Q(1/Q^k-1))^2 & ; \quad Q \neq 1, \end{cases}$$

$$(31)\quad dL_1/dQ = \begin{cases} 0 & ; \quad Q = 1 \\ m(kQ^{k-1}-(k-1)Q^k-1)/(Q^k-1)^2 & ; \quad Q \neq 1. \end{cases}$$

Suppose now that the floor manager follows the immediate feeding procedure after the first transfer lot, which is calculated as per Q^a. This procedure may be easier to implement if the machines are close, since the manager only needs to know L_1. In this case only the deviation in L_1 counts, and we have

$$(32)\quad dMS/dQ|_{\text{compounded error}} = |dL_1/dQ|\begin{cases} T_2(m-L_k^a)/L_1^a & ; \quad Q^a < Q^o, \\ T_1 & ; \quad Q^a > Q^o. \end{cases}$$

Again we plug the value of $|dL_1/dQ|$ from (31). Note that the L_j^a values here are the ones calculated using Q^a, and are analogous to the L_j^o values in (26).

Finally, by taking the partial derivatives of Q by T_1 (i.e., 1) and T_2 (i.e., $-1/T_2^2$), and multiplying them by (29) or (32), we get a measure of the sensitivity of the model to errors in estimating T_1 and T_2.

By observation of (29) we see that unless Q = 1 the penalty for overestimating Q is not equal to the penalty for underestimating it. If we wish to estimate Q in such a manner that the expected penalty will be minimized, we have to use an estimator where the ratio between the probability of under- and overestimating Q will equal the ratio between the over- and underestimation penalties (e.g., see [8]).

6. Modifying the Algorithm for Application

We have solved (P1) analytically for two machines, under the implicit assumption that we have one batch (or job), and we start at time zero with both machines not set up yet. It is important to discuss other applications which our basic algorithm can handle as well. The first generalization we may wish to consider (apart from the generalization to several machines which is beyond the scope of this paper) may be to a situation where several batches (of different products) need to be processed on the same pair of machines.

If we limit ourselves to the flow shop case, i.e., all batches start on M_1 and continue to M_2, and assume the order of the batches is given, the algorithm may be applied without too many further modifications. In fact, the only way a preceding batch may influence the next one is by imposing bounds, similar and additional to SU_1 and SU_2, on the times M_2 will be available.

It is important to define our objective for this case before we proceed with the analysis. The simplest objective function to deal with, is to finish the last batch as soon as possible. Indeed we start with this case. (An interesting question in this case is whether or not Johnson's Rule is still optimal. Be that as it may, our assumption is that the order of the batches is given. In practice, we could either use the rule, or use the order defined by the promised due dates.)

As for M_1, we may safely assume that it is always busy -- either processing a batch, or being set up for the next one -- until the last batch is finished on it. This assumption, is equivalent to starting activities at the "early start" under PERT/CPM. For M_2, by choosing the "late start" policy instead, we may assume that it will also be busy continuously. In other words, once the optimal finishing time is determined, there is no reason not to schedule M_2's activities backwards from that time. Having done that, our problem becomes very similar to

the single batch problem. The only major difference is that we have to allocate the budget among the various batches.

One possible way to allocate the budget to the batches is by dynamic programming. We omit the details of this method here. Instead, we discuss a method based on the optimizing procedure and linear programming. We prefer the latter due to the wide availability of LP codes. It is also easier to adapt to another objective function discussed below.

Assume for a while that the optimal finishing time is known, or we have a close super-optimal value for it. Then we can actually use the optimizing procedure as described in Section 4. By doing this we will also be able to determine the transfer lots for all the batches. But in order to find the approximate finishing time, it's a quite straightforward procedure to use a search method such as the one discussed in Section 4. It remains to find a close super-optimal solution -- which we proceed to do by LP. Let:

- n be the number of batches to be processed,
- $T1_i$, $T2_i$, $SU1_i$, $SU2_i$, C_i , TT_i and m_i denote T_1, T_2, SU_1, SU_2, C, TT and m for batch i (C_i and TT_i may or may not vary with i),
- $FT1_i$ be the time batch i finishes on M_1,
- $ST2_i$ denote the time batch i starts on M_2,
- $y_{i,j}$ indicate if we use j (or more) extra transfers for batch i,
- $MG_i(j)$ be the marginal gain by the j^{th} transfer for batch i.

Then ST2 and y are our decision variables; MG is calculated as per (10), with the appropriate values for batch i (note that MG is defined for $j \geq 2$); since M_1 does not idle, FT1 is simply

(33) $$FT1_i = \sum_{j=1,i}(SU1_j + m_j T1_j).$$

And we are ready to formulate the LP model, as follows:

(34) $$\min \{Z = ST2_n + m_n T2_n\},$$

subject to:

(35) $\sum_{i=1,n\ j=1,m_n} y_{i,j} C_i \leq B,$

(36) $ST2_1 \geq SU2_1,$

(37) $ST2_{i+1} \geq ST2_i + m_i T2_i + SU2_{i+1} \quad ; \quad i = 1, \ldots, n-1,$

(38) $ST2_i \geq FT1_i + TT_i - \sum_{j=1,mi-1} y_{i,j} MG_i(j+1);$ all i,

(39) $y_{i,j} \leq 1 \quad ;$ all i, j,

(40) $y_{i,j} \geq 0 \quad ;$ all i, j.

This is similar to the LP approach for crashing in PERT/CPM [5, pp. 331-334]. The main difference is that we use a convex crashing cost scheme, (38), instead of a linearized one. The convexity of MG, as per Theorem 1, assures us that the transfers will be introduced in a feasible order; e.g., a third transfer for a batch will not be indicated before the second one is fully utilized (with y = 1). We still face two theoretical problems: (i) to ensure that MG is indeed monotone decreasing, we must allow fractional items to be transferable (otherwise, Theorem 1 does not apply); and, (ii) the linear programming solution may specify fractional transfers as well. (Ideally y should be a {0,1} variable, but this would take us into the realm of integer programming.) Therefore, using LP here can yield a super-optimal result. This is where the optimality procedure comes in.

If we accept the limitations of the LP model, we may use it with a more elaborate objective function as well. An obvious choice may be to minimize a weighted sum of several finishing times instead of that of the last batch alone. A more interesting objective function is a **"Just-In-Time"** objective function, which is well within the power of the LP model, as follows.

Assume that each batch has a due date, DD_i, associated with it. Delivering after DD_i will cause a penalty of L_i per time unit. Finishing production too early is also discouraged, because it implies holding costs of H_i per time unit.

(In practice L_i may be considerably larger than H_i.) If we define $FT2_i$ as the finishing time of batch i, then

(41) $\quad FT2_i = ST2_i + m_i\, T2_i \quad ;$ all i.

Letting LT_i denote the lateness of the batch, and ER_i its earliness, then

(42) $\quad LT_i - ER_i = FT2_i - DD_i \quad ;$ all i.

(41), (42) are new linear constraints which we may add to the LP formulation. LT_i and ER_i are our new decision variables, and we constrain them to be non-negative as well. Our new objective function will simply be

(43) $\quad \min \{Z = \sum_{i=1,k}(LT_i L_i + ER_i H_i)\}.$

In this case, we may improve Z by allowing M_1 to operate intermittently. We leave the details of the necessary changes required in the formulation for that purpose to the interested readers. We should also mention that this version may be run with or without the budget constraint. Yet another interesting option is to add the transferring costs to the new objective function, and let the LP solution optimize the transferring expenditure along with the delivery dates.

It remains to discuss the heuristic for applying the two machine model we developed to several consecutive machines. Here we introduce a simple greedy heuristic. Let $MG_{i,i+1}(k)$ denote the marginal gain for the pair (i,i+1) by the k^{th} transfer, as calculated by (10) with $Q = Q_i$. Theorem 1 suggests that it is at least nearly optimal to allocate the budget to the parts of the problem by listing and sorting all the marginal gain per dollar values, i.e., $MG_{i,i+1}(k)/C_i$, for the various pairs and k values, and include those transfers which yield the highest gain, as long as the budget allows it. This makes it possible to optimize the budget itself by adding transfers from the top of the list as long as the extra expense is justified. For an analytic solution of this problem by dynamic programming, see [9].

7. Conclusion

To conclude the paper let us discuss related issues which require further research. Excluding the issues already mentioned above.

Perhaps the most important related problem is the implication of the model on the regular job shop scheduling problem. This case is much more complicated than the flow shop case. Here our batch is probably one of many different job orders competing for resources at the same time. Ideally we would like to schedule all jobs on all machines while taking into account transfer lots and do it optimally. This is a very tall order indeed, since by introducing the transfer batch issue, we've complicated the mathematical model considerably, and it was intractable practically to begin with.

In practice, we may use a simple approach for this issue. For instance, sequence the activities first, and then, while scheduling them as per the sequence, see about transfer lots. If the best sequence is such that no overlapping can be achieved -- e.g., if all machines have long queues, and every job has to wait -- then we may be actually better off without partial transfer lots. On the other hand, if we operate our plant efficiently, the only machines which should be allowed to accumulate considerable queues are bottleneck machines (to avoid starving them). Therefore, it is highly likely that transfer lots will be useful after all. Furthermore, if a bottleneck resource is starved, we should probably use the method while processing items to feed it.

Another important issue is to address the stochastic nature of the real manufacturing environment. For instance, optimizing the estimator for Q (as mentioned briefly above), calculating the distribution of the makespan under this choice, what time buffers should be utilized, and so on.

REFERENCES:

[1] Baker, Kenneth R., Lot Streaming to Reduce Cycle Time in a Flow Shop, Working Paper #203, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH 03755, June 1987.

[2] Chase, Richard B. and Nicholas J. Aquilano, Production and Operations Management, 4th Edition, Richard D. Irwin, Inc., Homewood, Illinois, 1985.

[3] Goldratt, Eliyahu and Robert E. Fox, The Race, North River Press, Box 241, Croton-on-Hudson, 1986.

[4] Graves, Stephan C. and Michael M. Kostreva, Overlapping Operations in Material Requirements Planning, Journal of Operations Management-Special Combined Issue with APICS, 6, #3, 1986, pp. 283-294.

[5] Lee, Sang M., Lawrence J. Moore and Bernard W. Taylor, Management Science, 2nd Edition, Wm. C. Brown Publishers, Dubuque, Iowa, 1985., pp. 330-334.

[6] McLeavey, Dennis W. and Seetharama L. Narasimhan, Production Planning and Inventory Control, Allyn and Bacon, Inc., 1985.

[7] Orlicky, Joseph, Material Requirements Planning, McGraw-Hill, New York, 1975.

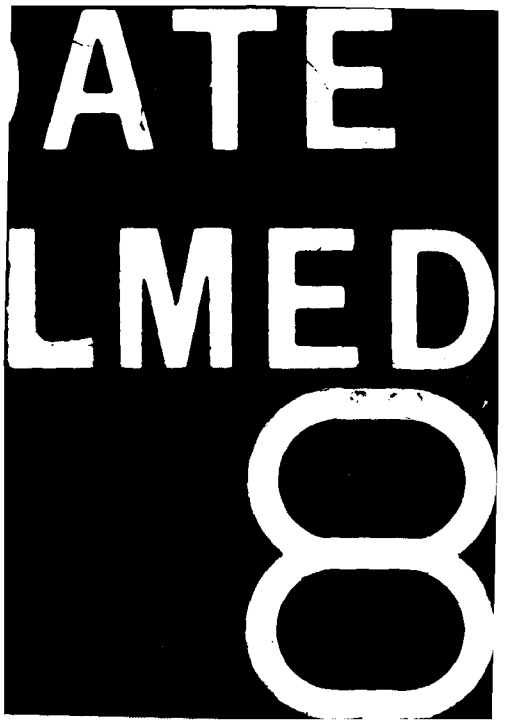
[8] Ronen, Boaz and Dan Trietsch, Optimal Scheduling of Purchasing Orders for Large Projects, Working Paper, October 1987.

[9] Trietsch, Dan, Optimal Transfer Lots for Batch Manufacturing On Several Machines, Working Paper, February and November 1987.

[10] Vollmann, Thomas E., "OPT as an Enhancement to MRP II," Production and Inventory Management, 2nd Quarter, 1986, pp. 38-46.

Distribution List

Agency	No. of copies
Defense Technical Information Center Cameron Station Alexandria, VA 22314	2
Dudley Knox Library, Code 0142 Naval Postgraduate School Monterey, CA 93943	2
Office of Research Administration Code 012 Naval Postgraduate School Monterey, CA 93943	1
Library, Center for Naval Analyses 4401 Ford Avenue Alexandria, VA 22302-0268	1
Department of Administrative Sciences Library Code 54 Naval Postgraduate School Monterey, CA 93943	1
Dan Trietsch Code 54Tr Naval Postgraduate School Monterey, CA 93943	40


ATE
LMED
8